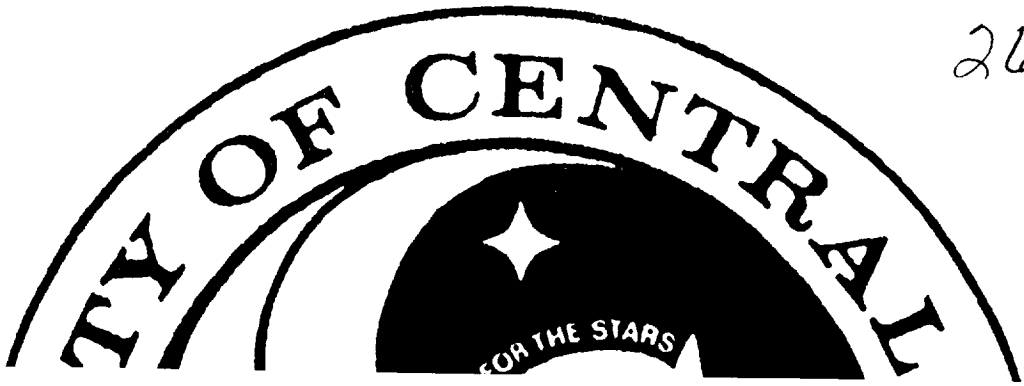
NASA-CR-193779



FINAL REPORT

University of Central Florida
Grant: NAG10-0042

AUTOMATED KNOWLEDGE GENERATION

FINAL REPORT

Submitted to:
National Aeronautics and Space Administration
John F. Kennedy Space Flight Center
Grant Number NAG10-0042

August 1990

Harley R. Myler Avelino J. Gonzalez
Department of Computer Engineering
University of Central Florida

College of Engineering Technical Report 90-1623202/3

Table of Contents

1.0 Introduction

The Automated Knowledge Generation system was proposed to the NASA Kennedy Space Center's Advanced Projects Office in August of 1987 as a possible means of solving the knowledge engineering problem associated with the creation of data bases for model-based reasoning systems. Computer Aided Design (CAD) databases were identified by NASA as a potential source of information for an automated knowledge generator (AKG) program that would convert CAD representations to model-base representations for use by automated reasoning programs. NASA funded an earlier study to convert CAD representations using a simple symbolic translation scheme; however, it was determined that the requisite functional knowledge required by the automated reasoning system targets could not be elucidated by a translator. The proposal offered by the University of Central Florida (UCF) was to construct a system that would use techniques of relaxation labelling borrowed from machine vision research to solve the problems of CAD database conversion to knowledge bases. In September of 1987, NASA awarded Drs. Avelino Gonzalez and Harley Myler of the UCF Computer Engineering Department a three-year grant to study and construct an Automated Knowledge Generation system.

This document gives a brief description of the AKG system, summarizes the results of research into the AKG system in the final year (1989-90), and discusses future research issues related to AKG. For a more complete description of the AKG system and details of it's theory and operation, the reader is directed to the following documents available from the University of Central Florida Library, 2500 University Blvd., Orlando, Florida, 32816:

- Functional Conflict Resolution in Automated Knowledge Generation, Ph.D. Dissertation, Massood Towhidnejad, August 1990.
- The Use of Constraints to Represent Process System Knowledge in Automated Knowledge Generation, MS Thesis, Frederick D. Mackenzie, May 1990.
- A Mega-heuristic Approach to the Problem of Component Identification in Automated Knowledge Generation, MS Thesis, Robin Rouch Kladke, December 1989.

In addition to the above, a number of papers have been published that describe various aspects of AKG. Copies of these publications can be found in Volume II of this report. Consult the bibliography at the end of this document for publications important to AKG in general, specifically the papers discussing the NASA Knowledge-Based Autonomous Test Engineer (KATE) and model-based reasoning systems in general.

2.0 Automated Knowledge Generation

The UCF AKG system is an automated reasoning tool designed to convert schematic descriptions of process control systems into model-base representations for use by automated diagnostic and control reasoning systems. Relational or hierarchical databases of Computer Aided Design systems may be accessed for input data to AKG, or the graph descriptions can be constructed manually from drawings. Figure 1 illustrates the data stream to AKG. As can be seen from the diagram, AKG requires only a Component list and a To-From connectivity list between the components.

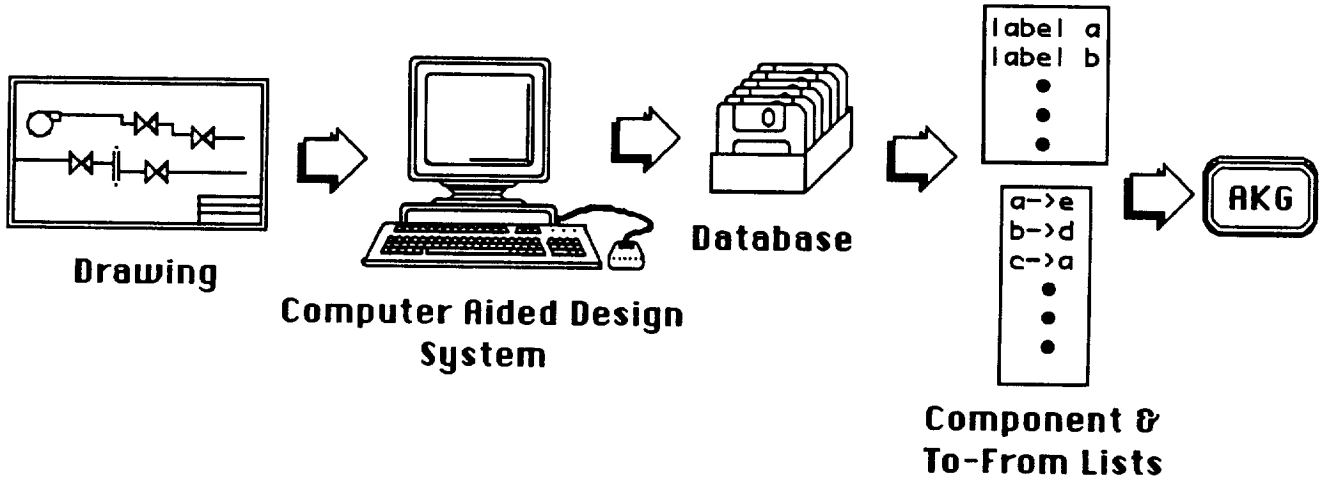


Figure 1. AKG Data Stream

The component labels, ideally, describe the component clearly and completely. For example, *2-Way Butterfly Valve* is a clearer description of a component than the abbreviation *But-fly vlv* and is more complete than just *2-Way Valve*. Any of these labels is better than a non-descriptive label such as *A213450* that simply indicates a part number from a parts list. The labelling approaches used on both CAD and manually generated drawings vary widely and cannot be assumed to contain all the information necessary to derive complex model-bases. In order to adequately label the components derived from a drawing of a process system more information must be garnered. The AKG system does this by propagating information through the component graph using natural connectivity constraints. AKG must have seed data that is both true and consistent. This data must come from either the input data (labels) or from the human operator. Once AKG has seed data, it can begin the process of propagating known labels to unknown labels--what is referred to as *resolving*.

The AKG program follows good software engineering design methods and is highly structured and modular. Structured programming allows a complex

system to be easily understood in terms of function and data flow while modularity allows easy modification and configuration management control. The AKG system diagram is shown in Figure 2. AKG consists of a number of processes that each perform a specific and important function in the creation of a knowledge base. The entire system is written in Common Lisp and runs on a Symbolics 3600 series AI workstation or equivalent.

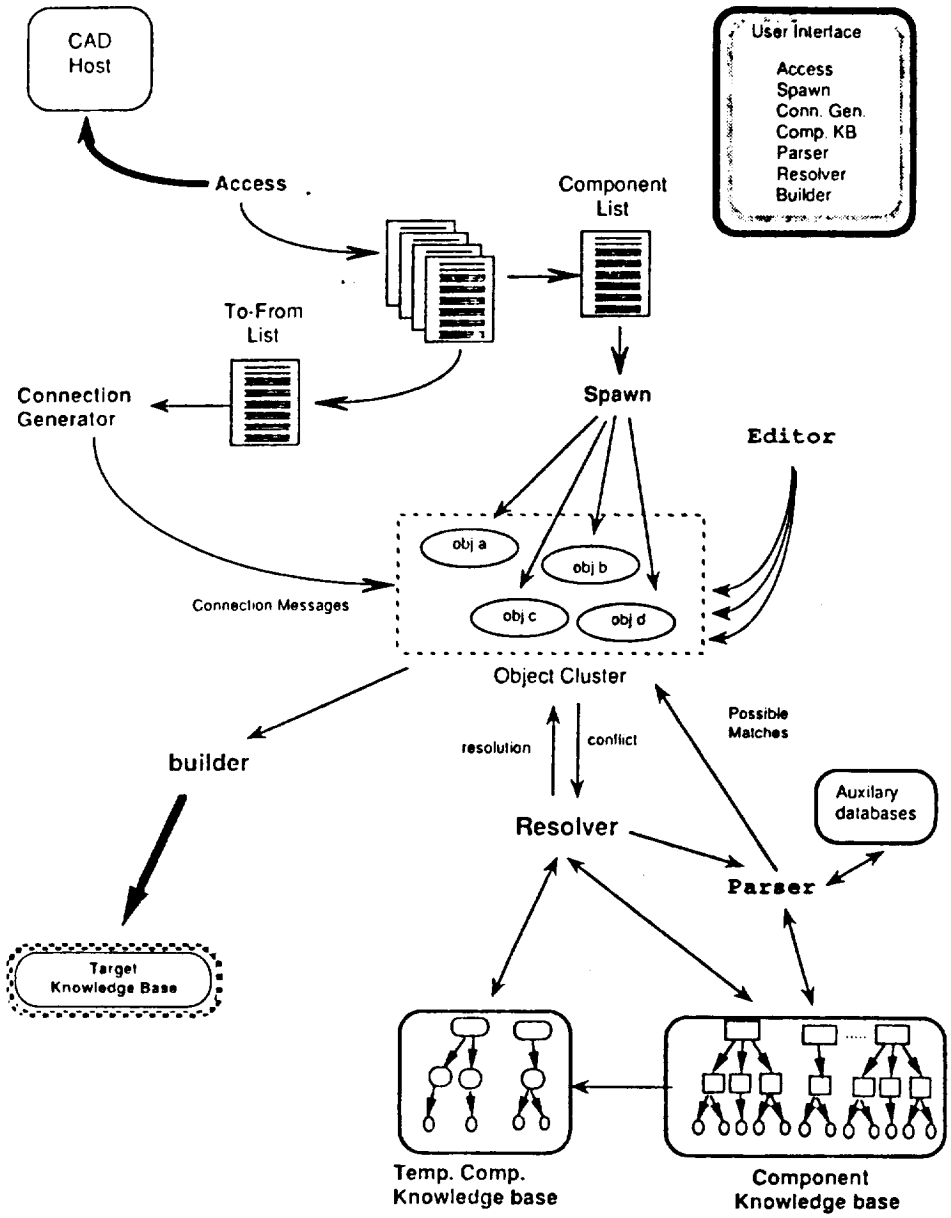


Figure 2. AKG System Diagram

Data enters the AKG program in the form of two ASCII files, **compoc.dat** and **tofromc.dat**. These names are not shown in the diagram. The files can be generated by a CAD system or by hand.

The **compoc.dat** has the following format:

Component Name	Component Description	Units
PW #3	Power Supply	VDC
OPAMP #2	OPERATIONAL AMPLIFIER	
R3	RESISTOR	OHM
R4	RESISTOR	OHM
R5	RESISTOR	OHM

The **tofromc.dat** has this format:

Component Name	Connecting Point	Component Name	Connecting Point
PW #3	+	R3	a
PW #3	-	R4	a
R3	b	OPAMP #2	I
R4	b	OPAMP #2	J
R4	b	R5	a
OPAMP #2	O	R5	b

These files are read into AKG by its *ACCESS* process and converted into two global lists, the *component-data-list* and the *to-from list*. The *component-data-list* is used by the SPAWN process to create a unique software object for each of the components and this object set is called the object cluster. A third process, the Connection Generator, uses the *to-from list* to provide connectivity information to the object cluster. At the completion of these three processes, AKG has stored in memory an *object-oriented* representation of the original process system drawing as a sparsely connected graph. The object-oriented representation allows for easy manipulation of the graph and this capability is essential to the process of determining an adequate set of models for the process system.

The internal representation used for each of the components in the object-cluster is given below:

```
(object PUMP
   (type SOURCE)
   (inputs ((ELECTRICAL-SOURCE ANALOG-VALUE) (FLOW-INPUT
            ANALOG-VALUE) (PRESSURE-INPUT ANALOG-VALUE)))
   (outputs (FLOW-OUTPUT ANALOG-VALUE)))
   (output-functions (FLOW-OUTPUT (IF (AND ELECTRICAL-SOURCE
            FLOW-OUTPUT) (SQRT (/ (- MAXIMUM-PRESSURE
            PRESSURE-INPUT) (+ PUMP-CV TOTAL-CV)))
            0.0)))
   (parameters ((PUMP-CV 15.4) (MAXIMUM-PRESSURE 6.5)))
   (strong-constraint (TYPE-OF-OUTPUTS TYPE))
   (normal-constraint (UNITS))
   (weak-constraint (RATING DELAY))
   (delay 2)
   (units (FLOW-INPUT GPM)))
```

Note that this representation is frame-like. The frames of the object-cluster, when complete, are translated by the *BUILDER* process into the data representation required by the automated reasoning system target. The *BUILDER* is nothing more than a formatter and is easily modified to output data to any model-based reasoning target system if the knowledge-base consists of component representations mapped directly to components of a system drawing. AKG cannot accommodate specialized functions that are not present on the system drawing. AKG can create pseudo or meta-components† if the objects are present in the input data (compoc.dat and tofromc.dat files). AKG cannot resolve pseudo or meta-components if their descriptors are not present in the component database.

The process of converting the graph component label nodes into a knowledge base begins with a table-lookup type of operation. The process is essentially a matching operation between the labels and known components in a database. The matching is performed by a lexical analysis of each component label by the *PARSER* process. The PARSER checks the label against labels stored in the COMPONENT KNOWLEDGE BASE (CKB). The CKB consists of a hierarchial tree of objects beginning with a small number of root nodes that represent generic object classes such as valves, transducers, etc.. The nodes decrease in generality until a specific component is represented at the leaves, for example, a *manual ball valve*. Each component in the CKB has a complete functional description that is retrieved and stored temporarily within AKG. After the object cluster has been parsed against the CKB, the conversion process is finished--unless there are

† components not represented by a physical object; *e.g.*, flow or pressure.

unresolved components. In this case, AKG uses it's RESOLVER module to attempt to determine the nature of the unknown objects using constraint propagation.

The user interface to AKG uses the bit-mapped graphic windowing capabilities of the Symbolics to provide a simple operating environment from which to generate knowledge bases. The process is interactive and requires human attendance; however, AKG was designed to be operated by a domain expert, in this case the process engineer. Section four of this report *Using the AKG System* discusses a knowledge base generation session with AKG on a typical process control system.

3.0 Accomplishments

The primary accomplishment of the final year of research and the completion of the grant was to finish the Automated Knowledge Generation system as described and discussed within reports over the last three years. The system is fully functional and capable of converting CAD data automatically into knowledge-bases with minimal human involvement. There are a set of caveats that accompany the previous statement and these are:

- CAD system data must be in the form of two lists, one of component labels and one of component interconnectivity.
- The automated reasoning system target must:
  - be frame-based, and
  - assume one-to-one correspondence between components in the system acted on and the model-base.

AKG, at present, is written in Symbolics™ Common Lisp and may only be run on Symbolics™ machines or emulators.

A number of tests have been run to verify the operation of AKG. The tests include process systems and special experiments derived from digital circuits. These are discussed at length in the dissertation listed in Section 1.0. The limitations to AKG are bounded by the amount of information contained in the Component Knowledge Base and the representation required by the target reasoning system. In Section 4.0, we discuss a primary source of AKG incompatibility, the use of pseudo-objects to describe conceptual entities that are not inherent to a process drawing.

4.0 Pseudo-Objects Analysis

Pseudo-objects are objects or frames that represent engineering concepts (flow, resistance, pressure, temperature, etc.) of specific components or a group of components. They may even form a network that does not necessarily correspond to the way in which components are physically connected, but that interacts with such a network of interconnected components.

In the case of KATE, pseudo-objects are needed to completely describe system functionality and are used to mediate the relationships between structurally remote components. KATE does not expect to find a direct connection between two physical objects or components that are remote structurally. In addition, pseudo-objects are also used to localize complex computations such as collecting all the information about a specific concept or property, making a calculation based on the information and passing the result to a physical component (usually a measurement component) or to another pseudo-object for further calculations.

In order to add pseudo-objects to a model base of a system, the complete structure of the system must be known. The entire system structure is needed, because, as mentioned previously, even though a specific transfer function is associated with each component, when all the components of a system are connected, the properties from components upstream and downstream a specific component affect such component's transfer function. Thus, pseudo-objects take care of the relationships among the engineering concepts that arise from the assembly and connection of the whole system and its actual functioning. Consequently, to derive pseudo-objects for a system, some type of network analysis is needed. The flow of the system is analyzed to produce the network of pseudo-objects.

It seems that pseudo-objects may be avoided by allowing structurally remote and functionally related objects to be directly connected in the model and by allowing multiple transfer functions in the object of a single component. According to (Scarl et al., 1987), the engineering concepts represented by pseudo-objects could be left implicit in the objects of the physical components, but it is usually conceptually cleaner to represent them separately. As discussed before, they localize complex computations and allow for the representation of global engineering concepts for the entire system as well as for subsystems. Besides, the use of pseudo-objects eases the process of functional inversion in KATE by avoiding objects with multiple transfer functions.

Examples of the clarified representation provided by pseudo-objects are observed in the model of the Red Wagon System constructed for KATE. In that

model, separate pseudo-objects were used to calculate the admittances of several subsystems like the all-pumps circuit (four pumps with their associated valves and flow sensors), the feedback area (a valve and a flow sensor) and the fill circuit (two valves and two flow sensors). Each of these pseudo-objects obtained the values for their inputs from other pseudo-objects that calculate admittances for smaller subsystems. The inputs of this network of pseudo-objects come from physical objects. In this way, pseudo-objects kept this network of calculations out of the physical component objects. At the same time, subsystems (subgroups of components within the entire system) have their properties represented making it possible to recognize subsystems during diagnosis as the cause of failure and to descend into these subsystems and attempt to isolate a culprit assuming single-point failure (see NASA AI Lab Memos listed in the Bibliography).

The inference engine in an automated reasoning model-based system, in this case KATE, has several functions. It simulates the behavior of the real system by means of the model, compares the values calculated in the model with those measured in the real system, and then searches upstream and downstream for the cause of any disagreement between these values. Such discrepancies are used for automatic control error signals or may be evaluated to indicate the malfunctioning of one or several components. When looking for a faulty component, KATE goes from measurements (that indicate disagreement with calculated values in the model) to commands, performing the process of functional inversion. When going back through the objects to the commands, the transfer function of each object is inverted to get values for the input(s) given the output. With multiple transfer functions, one of the problems to face would be the many more paths to explore in the network of objects because of the multiple functions to invert in a single object. Consequently, objects with multiple transfer functions (often referred to as multiple-output objects) are avoided because they complicate the process of functional inversion that KATE performs to locate faulty components.

The above arguments explain in some way the presence of pseudo-objects in KATE's models and give very general guidelines for their use, but from the AKG standpoint it is important to determine what it takes to automate the generation of such pseudo-objects.

The issues concerning the automated generation of pseudo-objects mainly relate to one of the following requirements. First, a standard, more specific procedure or mechanism for the manual generation of pseudo-objects is needed. Second, in the case of the calculation of some global engineering concepts represented by pseudo-objects, the network analysis required to produce their transfer function must be represented in some useable way.

The NASA/Boeing team has been successful in the creation of models for real systems, models that have included a great deal of pseudo-objects (Belton et al., 1990). The LOX, Purge Demo, and ECS systems were successfully modelled with the original KATE frame representation and the Red Wagon System was the first one to be represented as a working model with the most recent KATE frame representation format (NASA AI Lab, 1990), which is also being used for the creation of a model for the Water Tanking (ALO-H_2O) System. Since there could be many right ways of modelling a system, the team has not been able to be consistent in the creation of these knowledge bases, thus causing difficulty in the establishment of specific guidelines for knowledge base (and pseudo-object) generation. Representation inconsistencies like the following have been found mainly from observations made in the models of the Red Wagon System (RWS) and the Water Tanking System (WTS). Some components are modelled with frames containing different slots in each one of these systems; also, other components are modelled with a different number of pseudo-objects in each system. For example, a simple flow meter is described in the RWS by a frame with the following slots:

```
(inputs fluid-in flow-in)
(outputs elec-out flow-out)
(output-functions (elec-out (/ flow-in 1.07))).
```

In the WTS, however, these slots are shown for a similar flow meter:

```
(inputs flow-in)
(outputs flow-out)
(output-functions (flow-out flow-in)).
```

Some other components with different slots from one system to the other are the pump, the vehicle tank, and the storage tank. In the WTS, the pump and the motor that drives it are modelled separately; meanwhile, in the RWS, the motor is not modelled, but the pump is modelled with its admittance and pressure considered separately as pseudo-objects.

If these apparent inconsistencies in representation respond to the needs of the particular system, this confirms the need for a deeper study of the issues involved in component representation and the particularities and varying needs of different types of systems and various arrangements of the same system. The representation of transfer functions and pseudo-objects is too domain-specific and system-specific. For example, the same valve might need different transfer functions for the same property and, consequently, different pseudo-objects

depending on its position in a system. Also, depending on the domain involved and the measurements in the system, a component might need a transfer function for a different engineering property from the group of properties that can possibly be modelled in that component.

At some point, there was some disagreement among KATE developers about having either multiple-output or single-output objects (Belton et al., 1990). The choice between these two dictates the degree to which pseudo-objects are going to be used since the multiple-output approach require less use of pseudo-objects than the single-output approach. Multiple-output objects allow multiple transfer functions and the pseudo-objects are relieved from the task of representing the additional engineering properties. The team suggested in one occasion (Belton et al., 1990) the assumption of objects with a single transfer function and the identification of a unique concept different from all real outputs to be the only output for each component. The transfer function would determine the value of this unique output that would unite and relate all the component's real outputs, which were going to be transferred to separate pseudo-objects. The identification of this unique concept would require a good amount of knowledge of component design and the behavior of components in different systems.

In the RWS overview diagram shown in figure 3, the area that includes the pumps, the manual valve, and the storage tank is marked. The group of frames

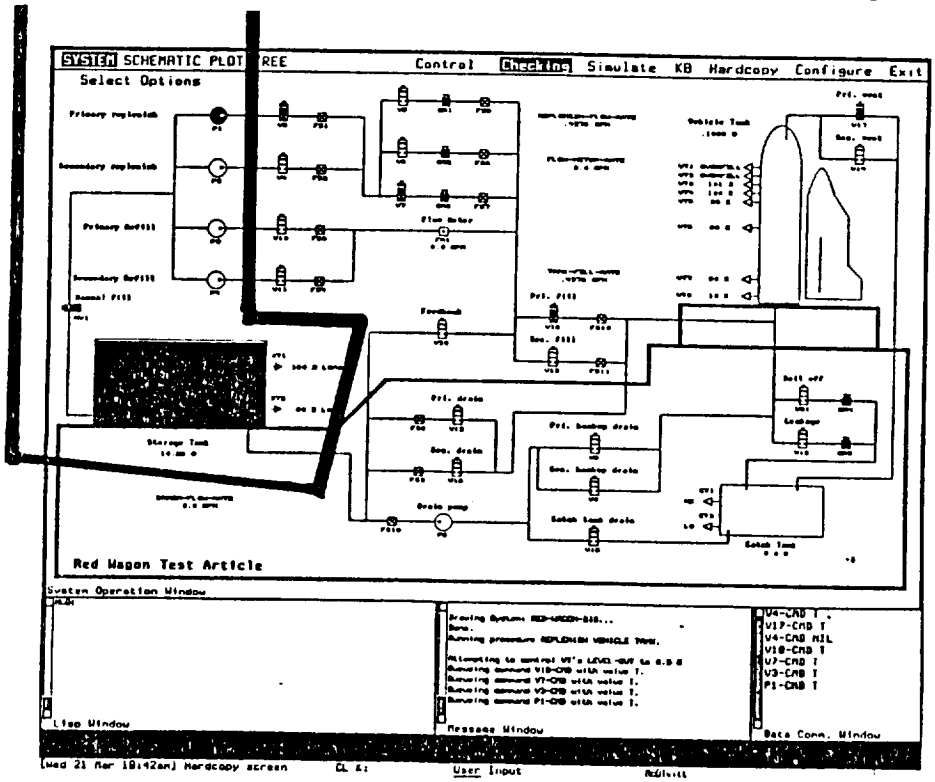


Figure 3. Red Wagon System (RWS) Overview

of the RWS knowledge base that represents this small area is illustrated in figure 4. Each box in figure 4 represents the frame of a physical component, each circle represents the frame of a pipe, and each ellipse represents the frame of a pseudo-object. The arrows indicate the direction of the flow of information between object frames. In the RWS knowledge base, a single transfer function per frame was used. If more than one engineering property required a transfer function in the component, pseudo-objects are used. That is why components like the storage tank and the pumps are surrounded by pseudo-objects. If multiple transfer functions would have been allowed in the RWS knowledge base, there would be less pseudo-objects around the objects representing real components.

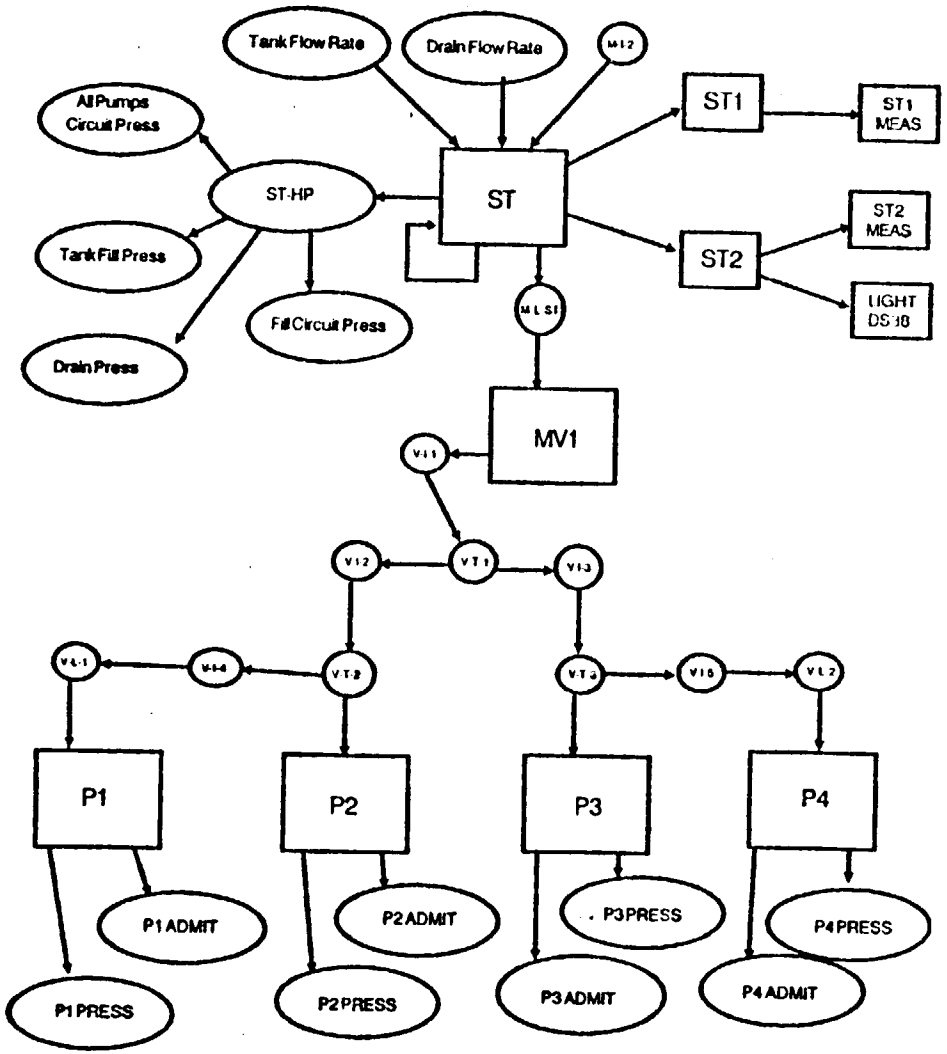
In order to produce the transfer functions for the pseudo-objects representing global engineering concepts, network analysis needs to be performed. For example, Charles Goodrich obtained the transfer functions for the admittances, pressures and flows of the Red Wagon System by reducing it to a network of electrical components (Goodrich, 1990). He considered the tanks as voltage sources and the pumps, sensors, pipes, and valves as resistors, each of which had an associated admittance. He used the equation

$$\text{flow} = (\text{pressure})^{1/2} * \text{admittance},$$

which reminds one of Ohm's Law. The need for this type of analysis requires a great deal of system design knowledge in the different domains handled by KATE.

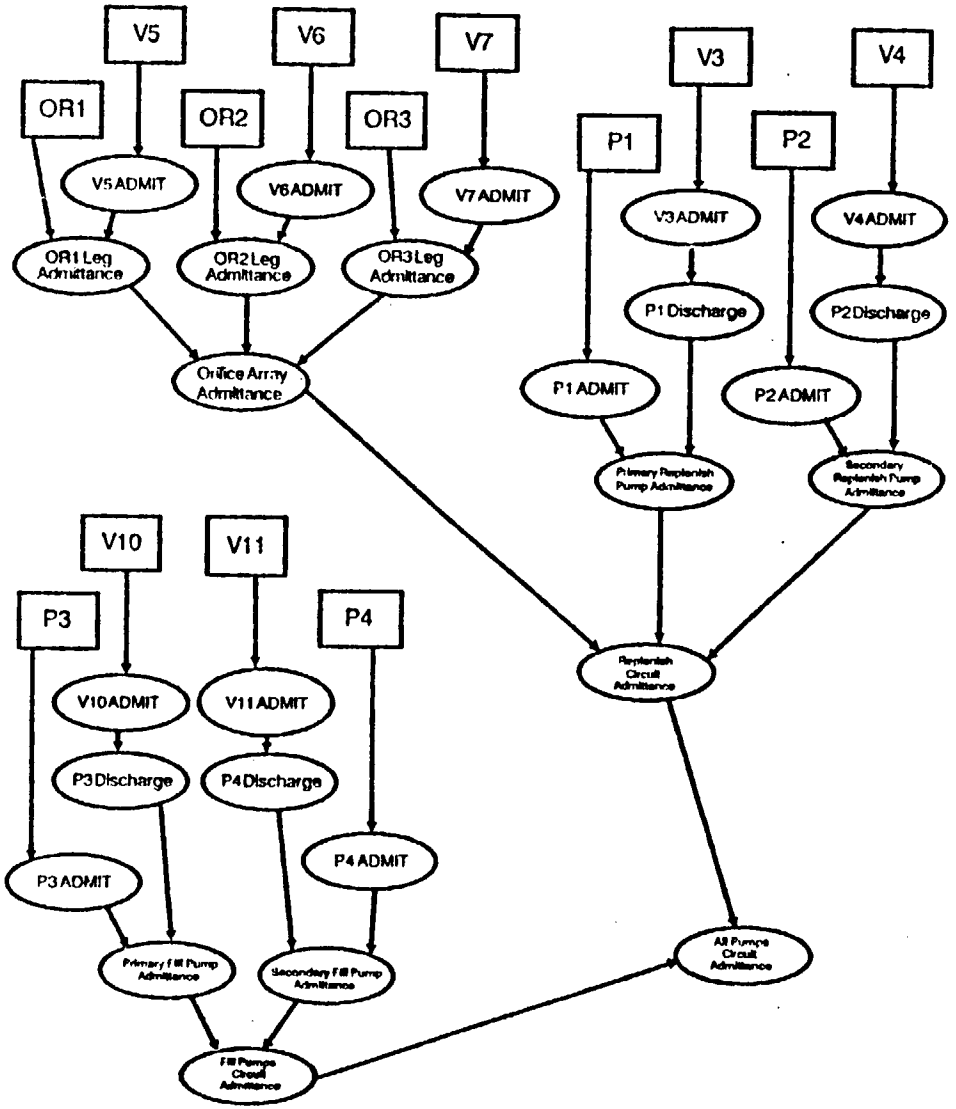
Object-oriented programming is a good technique for modelling systems for diagnosis and control. However, for the automated creation of pseudo-objects that require network analysis, a more global method of system representation is needed. This is because as indicated by (Borning, 1981), object-oriented languages generally emphasize a very localized approach to interaction, that is, an object interacts with other parts of the system only by sending and receiving messages to other objects that it knows about. For network analysis, the global view of the entire system is needed.

From this network analysis, KATE developers generate a complete network of pseudo-objects. For instance, this can be seen in the RWS knowledge base, in which some sections include complete networks of just frames of pseudo-objects. In figure 5, again, each box represents the frame of a physical component, each ellipse represents the frame of a pseudo-object, and the arrows indicate the direction of the flow of information between object frames, in this case, most of them, pseudo-objects. In this group of frames, the total admittance of the areas where the pumps are found is being calculated in a pseudo-object (called ALL-



(Box = physical component, circle = pipe, ellipse = pseudo-object.)
* EACH BOX, CIRCLE, AND ELLIPSE REPRESENTS A FRAME IN THE KNOWLEDGE BASE.
* CONNECTIONS AND ARROWS ILLUSTRATE INFORMATION FLOW DIRECTIONS.
* THIS IS A SECTION OF THE RED WAGON KNOWLEDGE BASE (red-wagon.lisp.287, pp. 5-7,9,17,19).

Figure 4. Psuedo Objects generated around physical component objects



(Box = physical component, ellipse = pseudo-object.)
* EACH BOX AND ELLIPSE REPRESENTS A FRAME IN THE KNOWLEDGE BASE.
* CONNECTIONS AND ARROWS ILLUSTRATE INFORMATION FLOW DIRECTIONS.
* THIS IS A SECTION OF THE RED WAGON KNOWLEDGE BASE (red-wagon.lisp.287, pp.21-30).

Figure 5. Network of Pseudo-Objects

PUMPS-CIRCUIT-ADMITTANCE) by adding admittances from two subsystems in the RWS that contain pumps, the fill pumps area and the replenish area. These subsystem admittances are being calculated in pseudo-objects (FILL-PUMPS-CIRCUIT-ADMITTANCE and REPLENISH CIRCUIT ADMITTANCE) that, in turn, obtained values from other pseudo-objects that calculate the admittances for other subareas (PRIMARY-FILL-PUMP-ADMITTANCE and SECONDARY-FILL-PUMP-ADMITTANCE in the case of FILL-PUMPS-CIRCUIT-ADMITTANCE). The sequence of pseudo-objects continues in this way until the values are obtained from the individual physical components in the subsystem (like the pump P3 and and the valve V10). The network does not stop here, however, since these pseudo-objects are, at the same time, being utilized by other pseudo-objects that calculate the values of other engineering properties needed by measurement devices.

The difficulty in generalizing the generation of pseudo-objects provokes the thought that there might be alternative approaches to the use of pseudo-objects. Pseudo-objects might be eliminated if some way to represent the system more globally forming modules (collection of parts or objects) out of those frequent arrangements of components that have some kind of common property or engineering concept that needs to be represented. Some ideas are described by (Davis, 1985) and (Borning, 1981). The modules would be a group of components and, at the same time, each would have a transfer function of its own describing a global property. The problem with this approach is that there might be an infinite number of arrangements of components to be represented.

A generalization of the procedures to manually generate pseudo-objects is needed to automate the process. However, this generalization could be possible only after actually modelling many systems of all kinds, which will allow an in-depth look at many different situations. Generalization from observations in a few KATE models or from interviews with members of the KATE developing team, especially when there is no consensus of opinion among them, is not feasible.

In order to automate pseudo-objects, a consistent procedure to implement them manually must be agreed upon first. The only guidelines available, which are found in (NASA AI Lab, 1990), are still too vague and leave ample room for a myriad of interpretations. Pseudo-objects have worked, but a standard for them has not been adopted. A deeper study of the issues involved in their implementations and the particularities of the multiple different systems that KATE may handle is another research topic altogether. The degree to which pseudo-objects are used will depend on the approach taken to represent functionality in the frame-based knowledge base. This choice of the approach will take into account the limitations and capabilities of KATE itself in terms of

the methods used by it to make pertinent diagnostic and control functions. This new research should lead to generalizations, if this is at all possible, that allow the creation of a more standard procedure to generate pseudo-objects that could be used in automation. Automation later will have its own issues, including those pseudo-objects that represent global system engineering concepts that might not be easily handled in an object-oriented environment and that might need a more global approach of representation.

5.0 Conclusion and Areas for Future Research

The Automated Knowledge Generation project has been a learning experience for both NASA and the University of Central Florida. The original goal of automating knowledge acquisition has been met, however, the system described has limitations that can be reduced with further research and development. We have identified two basic areas that show promise for new research efforts that build on the lessons learned from the AKG study. In addition to new research areas, the AKG system could be greatly enhanced by relatively simple operational improvements.

5.1 Operational Improvements to the AKG System

Translation of the AKG system from Common Lisp into C++ would greatly expand the number of computer platforms that the software would run on. The AKG system was designed using good software engineering practice such that all functions are self-contained and modular. The user interface, which requires a graphic windowing environment, is separable from the other AKG functions that process the CAD data and interface with the Component Knowledge Base. A window system such as X would be ideal for porting the user interface to and would further expand the range of machines capable of running AKG software.

The Component Knowledge Base (CKB) currently holds approximately 350 entries. These were determined from the experimental systems used during the project to test and verify AKG functions. If AKG were to be used on large, complex systems, the CKB would require expansion. This expansion would best be performed off-line and not by discovery. A possible option would be to automate the buildup of the CKB, however, it may prove easier to create a funded task to perform the work. In any event, the CKB represents a basic boundary on AKG ability to perform and any expansion of the CKB will improve AKG.

The use of parallel processors is becoming more frequent as newer, less expensive, and higher performance architectures become available. The search and constraint propagation processes that occur in AKG would lend themselves

to parallel processing approaches. The Parser essentially searches the CKB for each component in serial. This process would immediately improve with a parallel architecture since these searches are independent of component. Also, components with high confidence tend to be confined to localized areas making the constraint propagation process an excellent candidate for parallel processing.

5.2 Automated Design Data Capture

Design Data Capture addresses the problem of determining what methodologies and actions lead to good engineering design. The automation of this function is an attempt to capture information from humans that is both critical to design yet inherently transparent to observers. A primary approach to the problem involves capturing knowledge from designs as they are created during work being performed through the use of a CAD system. Automated Design Data capture can be viewed as an extension to AKG because of the ability of AKG to access and interpret CAD.

Research into Automated Design Data Capture has been identified as a critical area of AI research by both NASA Ames Research Center and Stanford University. The extension of tools such as AKG to aid in these studies is a natural direction for further research in the area.

5.3 Automated System Identification

A loose assumption often made by model-based reasoning systems that are constructed of component models is that the components are independent in behavior and that linear superposition of function holds. This was the situation with the KATE system. KATE assumes that a model, *vis a vis a component*, is independent and that all data streams in the target system are observable. The addition of pseudo-objects to the KATE functional inversion mechanisms solves this problem, however, as discussed in Section 4, a number of other issues become sensitized--specifically, the consistency and standardization of the model. AKG was designed to create model-bases with the assumptions above and cannot accommodate complex pseudo-object structures without modifications to the design approach. As an alternative, research into the development of an Automated System Identification program that combines functions of both KATE and AKG might be productive.

An Automated System Identification program would, like KATE, model the target system dynamically and control and diagnose in real-time. Like AKG, it would derive data from CAD representations, however, the identification functions would operate in real-time and functional parameters would be modifiable in

real-time. This program would be unlike KATE and AKG in that the system model would be global in scope and not rely on individual component models. Instead, the system would construct world models that describe physical functionality of the system in terms of flows, both electrical, fluid, energy and information. The checkpoints of the flow data would be prescribed by existing sensor data determined from the CAD drawings. The world models would be constructed from the collective behavior of the component elements in the same way that an engineer constructs a mental representation of the system he is working with. The system would be provided with an image scanning front end so that ambiguities in component identification would be minimized.

Bibliography

Andreae, P. M. 1984. Constraint Limited Generalization: Acquiring Procedures From Example. In *Proceedings of AAAI,* Los Altos: Morgan Kaufmann. 6-10.

Bainbridge, B. 1988. The Explicit Representation of Control Knowledge. *Approaches to Knowledge Representation, An Introduction, edited by G.A. Ringland, and D.A. Duce,* England: Research Studies Press LTD.

Balzer, R. M. 1973. A Global View of Automatic Programming. In *Proceedings of the IJCAI.* Menlo Park, CA: Stanford Research Institute Publishing Company. 494-499.

Barr, A., and E. A. Feigenbaum. 1981. Automatic Programming. *The Handbook of Artificial Intelligence, Volume II,* Reading, MA: Addison-Wesley Publishing Company, Inc. 297-305.

Beasley, W. G. 1986. Expert System Rules from CAD Database. In *Proceedings of American Control Conference,* Seattle: 146-151.

Bennet, J. S. 1985. ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System. *Journal of Automated Reasoning,* 1: 49-74.

Bible, W. 1988. Constraint Satisfaction from a Deductive Viewpoint. *Artificial Intelligence,* 35, no. 3: 401-413.

Biswas, P. K. 1990. An object-Oriented Knowledge Acquisition Tool. In *Proceedings of the 3rd Florida Artificial Intelligence Research Symposium,* Cocoa Beach, FL: n.p. 96-100.

Boose, J. H. 1984. Personal Construct Theory and the Transfer of Human Expertise. In *Proceedings of the AAAI.* 27-33.

Borning, A. 1981. The Programming Language Aspects of the ThingLab, A Constraint-Oriented Simulation Laboratory. *ACM Transactions on Programming Languages and Systems,* 3, no. 4 (October): 353-387.

Borning, A. 1977. ThingLab -- An Object-Oriented System for Building Simulations Using Constraints. In *Proceedings of the IJCAI.* Pittsburgh: Dept. of Computer Science, Carnegie-Mellon University. 497-498.

Borning, A., Duisberg, R., and Freeman-Benson, B. 1987. Constraint Hierarchies. In *Proceedings of Object-Oriented Programming Systems, Languages and Applications.* New York: Association for Computing Machinery. 48-60.

Carbonell, J. R. 1970. AI in CAI: An artificial intelligence approach to computer-aided instruction." *IEEE Transactions on Man-Machine Systems,* MMS-11(4): 1970.

Chou, H., J. F. Garza, and N. Ballou. 1986. Supporting a Database System on Symbolics LISP Machines. *IEEE Computer Society Technical Committee on Database Engineering,* 9, no 3: 17-22.

Cornell, M. 1987. The KATE Shell An Implementation of Model-Based Control, Monitor and Diagnosis. In *Proceedings of the First Workshop on Space Operations Automation and Robotics,* Houston: n.p. 134-140.

Davis, L. S., and T. C. Henderson. 1981. Hierarchical Constraint Process for Shape Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI 3, no. 3 (May): 265-277.

Dechter, R. and A. Dechter. 1988. Belief Maintenance in Dynamic Constraint Networks. In *Proceedings of the AAAI.* Los Altos: Morgan Kaufmann. 37-41.

Dechter, R., and J. Pearl. 1989. Tree Clustering for Constraint Networks." *Artificial Intelligence,* 38, no. 3: 353-366.

Dechter, R., and J. Pearl. 1988. Network-Based Heuristics for Constraint-Satisfaction Problems. *Artificial Intelligence,* 34, no. 1: 178-183.

Dechter, R. 1986. Learning While Searching in Constraint-Satisfaction-Problems. In *Proceedings of the AAAI.* Los Altos: Morgan Kaufmann. 178-183.

Dixit, V., and D. I. Moldovan. 1984. Discrete Relaxation on SNAP., *IEEE Transaction,* 637-644.

Dreyfus, H. L. 1979. *What Computers Can't Do.* New York: Harper & Row.

Duisberg, A. 1986. Animated Graphical Interfaces Using Temporal Constraints. In *Proceedings of ACM CHI 86.* 131-136.

El-Hawary. M. E. 1984. *Control System Engineering*. Reston, VA: Reston Publishing Company.

Fox, M. 1981. Reasoning With Incomplete Knowledge in a Resource-Limitted Environment: Integrating Reasoning and Knowledge Acquisition. In *Proceedings of IJCAI*, Menlo Park, CA: AAAI Press. 313-318.

Freuder, E. C. 1978. Synthesizing Constraint Expressions. *Communications of the ACM*, 21, no. 11: 958-965.

Gelman, A., S. Altman, M. Pallakoff, K. Doshi, C. Manago, T. C. Rindfleisch, and B. G. Buchanan. 1988. FRM: An Intelligent Assistant for Financial Management. In *Proceedings of the AAAI*. Los Altos: Morgan Kaufmann. 31-36.

Genesereth, M. R. 1984. The use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence*, 24. no. 3: 411-436.

Hamscher, W., and R. Davis. 1984. Diagnosing Circuits with State: An Inherently Underconstrained Problem. In *Proceedings of AAAI*, Los Altos: Morgan Kaufmann. 142-147.

Haralick, R. M., and G. L. Elliott. 1980. Increasing Tree Search Efficiency for Constraint Satisfaction Problem. *Artificial Intelligence*, 14, no. 3: 263-313.

Haralick, R. M., and L. G. Shapiro. 1979. The Consistent Labeling Problem: Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 1, no. 2 (April): 173-184.

Harel, D. 1987. *Algorithmics The Spirit of Computing*. Wokingham, England: Addison-Wesley Publishing Company.

Hashemi, S. 1989. PLant EXpert SYStem (PLEXES), A Model-Based Reasoning Approach to Plant Modeling. In *Proceedings of American Institute of Chemical Engineer (AICHE)*, Houston: n.p. 87-93.

Hashemi, S., L. J. Paterson, J. Somsel, R. E. Colley, and R. S. May. 1989. Application of PLEXSYS in Nuclear Power Plants: Technical Specifications Monitoring and Maintenance Management. In *Proceedings of Expert Systems Applications for the Electric Power Industry*, Orlando, FL, n.p. 68-74.

Hayes-roth, F., D. A. Waterman, and D. B. Lenat. 1983. *Building Expert Systems*. Reading, MA: Addison-Wesley.

Hewitt, C. 1971. "Description and Theoretical Analysis (Using Schemas) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot." Ph.D. dissertation, AI Laboratory, MIT.

Hummel, R. A., and S. W. Zuker. 1983. On The Foundations of Relaxation Labeling Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 5, no. 3 (May): 267-287.

Intellicorp. 1988. *Knowledge Engineering Environment, KEE, Users Manual.* Mountain View, CA: Intellicorp.

Intergraph. 1987. *Data Management and Retrieval System. Volume II, DIXD3310,* Huntsville, AL: Intergraph Corporation.

Jamieson, J. R., E. A. Scarl, and C. I. Delaune. 1985. A Knowledge Based Expert System for Propellant System Monitoring at the Kennedy Space Center. In *Proceedings of 22nd Space Congress,* Cocoa Beach, FL, n.p. 242-247.

Kahn, G., S. Nowlan, and J. McDermott, 1985. MORE: An Intelligent Knowledge Acquisition Tool. In *Proceedings of IJCAI-85,* Los Altos: Morgan Kaufmann.

Kasif, S. 1986. On the Parallel Complexity of Some Constraint Satisfaction Problems. In *Proceedings of the AAAI.* Los Altos: Morgan Kaufmann. 349-353.

Kemper, A., and M. Wallrath. 1987. An Analysis of Geometric Modeling in Database System. *ACM Computer Surveys,* 19, no. 1 (March): pp 47-90.

Ketonen, J. A. 1989. Toward Reasoning About Data. *AI Expert,* February, 44-49.

Klahr, P., and D. A. Waterman. 1986. *Expert Systems Techniques, Tools, and Applications.* Reading, MA: Addison-Wesley Publishing Company.

Ladkin, P. B. 1988. Satisfying First-Order Constraints About Time Interval. In *Proceedings of AAAI,* Los Altos: Morgan Kaufmann. 512-517.

Leler, W. 1988. *Constraint Programming Languages Their Specification and Generation.* Reading, MA: Addison-Wesley Publishing Company, Inc.

Levitt, D. 1984. Machine Tongues X: Constraint Languages. *Computer Music Journal,* 8, no. 1: 9-17.

Loveland, D. W., and M. Valtorta, 1983. Detecting Ambiguity: An Example in Knowledge Acquisition. In *Proceedings of IJCAI,* Los Altos: Morgan Kaufmann. 182-184.

Mackworth, A.K., and E.C. Freuder. 1985. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence,* 25, no. 1: 65-74.

Mackworth, A. K. 1977. Consistency in Network of Relations. *Artificial Intelligence,* 8, no. 1: 99-118.

Marcus, S., and J. McDermott, 1989. SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems. *Artificial Intelligence,* 39, no. 1. 1-37.

Marcus, S., J. McDermott, and T. Wang. 1985. Knowledge Acquisition for Constructive Systems. In *Proceedings of IJCAI-85,* Los Altos: Morgan Kaufmann.

Mathews. 1987. *Numerical Methods for Computer Science, Engineering and Mathematics.* Prentice-Hall.

McAllester, D. A. 1878. A Three Valued Truth Maintenance System. *Massachusetts Institute of Technology Artificial Intelligence Laboratory,* AI Memo 473, 1-24.

McDarmott, J. 1982. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence,* 19, no. 1: 31-88.

Menzies, T. 1989. Domain-Specific Knowledge Representation. *AI Expert,* June 36-45.

Mingruey, R. T., and S. N. Srihari. 1987. Modeling Connections for Circuit Diagnosis. In *Proceedings of the IEEE 3rd Conference on Artificial Intelligence,*

Minton, S. 1984. Constraint-based Generalization Learning Game-Playing Plans from Single Examples. In *Proceedings of the AAAI.* Los Altos: Morgan Kaufmann. 251-254.

Moher, R., and T. C. Henderson. 1986. Arc and Path Consistency Revisited. *Artificial Intelligence,* 28, no. 2: 225-233.

Morgenstern, M. 1984. Constraint Equations: A Concise Compilable Representation for Quantified Constraints in Semantic Networks. In *Proceedings of the AAAI.* Los Altos: Morgan Kaufmann. 255-259.

Nachtsheim, P. R. 1989. Solving Constraint Satisfaction Problem. *AI Expert,* June, 30-35.

NASA Kennedy Space Center, "Knowledge Base Generation Guide". *Internal Technical Report, NASA-KSC,* Florida, March 1989.

NASA Kennedy Space Center, "KATE Knowledge Base Generation Guide." *Internal Technical Report, NASA-KSC,* Florida, November 1989.

Nishihara, S., and K. Ikeda. 1985. A Constraint Synthesizing Algorithm for the Constraint Labeling Problem. In *Proceeding of 7th International Conference on Pattern Recognition.* Washington: Computer Science Press. 310-312.

Nishihara, S., and K. Ikeda. 1986. A Solution Algorithm for the Constraint Labeling Problem Using the Structure of Constraints. In *Proceeding of 8th International Conference on Pattern Recognition.* Washington: Computer Science Press. 198-200.

Ohlsson, S. 1983. A Constrained Mechanism for Procedural Learning. In *Proceedings of the IJCAI.* Los Altos: Morgan Kaufmann. 426-428.

Parsaye, K. 1988. Acquiring and Verifying Knowledge Automatically. *AI EXPERT,* May 48-54.

Pazzani, M. J. 1986. Refining the Knowledge Based of a Diagnostic Expert System: An Application of Failure-Driven Learning. In *Proceedings of AAAI,* Los Altos: Morgan Kaufmann. 1029-1035.

Pearl, J. 1986. Fusion, Propagation, and Structuring in Belief Networks." *Artificial Intelligence,* 29, no. 3: 241-288.

Price, K. E. 1985. Relaxation Matching Techniques - A Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI 7, no. 5 (September): 617-623.

Purdom, P. W. 1984. Solving Satisfiability with Less Reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI 6, no. 4 (July): 510-513.

Rendell, L. 1985. Substantial Constructive Induction using Layered Information Compression: Tractable Feature Formation in Search. In *Proceedings of the 9th IJCAI,* Los Altos: Morgan Kaufmann. 650-658.

Rich, E. 1983. *Artificial Intelligence.* New York: McGraw-Hill Book Company.

Rissland, E. L., and E. M. Soloway. 1981. Constrained Example Generation: A Testbed for Studying Issues in Learning. In *Proceedings of the IJCAI.* Menlo Park, CA: AAAI Press. 162-164.

Rit, J. 1986. Propagating Temporal Constraints for Scheduling. In *Proceedings of AAAI,* Los Altos: Morgan Kaufmann. 383-388.

Rosenfeld, A., R. A. Hammel, and S. W. Zucker. 1976. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-6, (June): 420-433.

Rumbaugh, J. 1987. Relations as Semantic Constructs in an Object-Oriented Language. In *Proceedings of OOPSLA,* New York: Association for Computing Machinery. 466-481.

Rychener, M. D. 1988. Research in Expert Systems for Engineering Design. *Expert Systems for Engineering Design,* Edited by Michael D. Rychener, San Diego: Academic Press INC.

Scarl, E. A., J. R. Jamieson, and C. I. Delaune. 1987. Diagnosis and Sensor Validation through Knowledge of Structure and Function. *IEEE Transactions on Systems, Man, and Cybernetics,* MMS-17(3): 165-169.

Seidel, R. 1981. A New Method For Solving Constraint Satisfaction Problems. In *Proceedings of the IJCAI.* Menlo, CA: AAAI press. 338-341.

Shaw, M. G. 1982. PLANET: Some Experience in Creating an Integrated System for Repertory Grid Applications on a Microcomputer. *International Journal of Man-Machine Studies, 17,* 345-360.

Smith, C. A., and A. B. Corripio. 1985. *Principles and Practice of Automatic Process Control.* J. Wiley.

Stahl, H., K. M. Ford, J. R. Adams-Webber, and J. Novak. 1990. ICONKAT: Integrated Constructivist Knowledge Acquisition Tool. In *Proceedings of the 3rd Florida Artificial Intelligence Research Symposium,* Cocoa Beach, FL: n.p. 90-95.

Stallman, R. M., and G. J. Sussman. 1977. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence,* 9, no. 2: 135-190.

Steinberg, L. I. 1987. Design = Top Down Refinement Plus Constraint Propagation Plus What? In *Proceedings of Man and Cybernetic,* Piscataway, NJ: IEEE Computer Service Center. 498-502.

Steele, G. L., and G. J. Sussman. 1978. Constraint. *Massachusetts Institute of Technology Artificial Intelligence Laboratory,* AI Memo No. 502.

Steels, L. 1983. Descriptions as Constraints in Object-Oriented Representation. In *Proceedings of the IJCAI.* Los Altos: Morgan Kaufmann. 395-397.

Stefik, M. J. 1981. Planning with Constraints (MOLGEN: Part 1); Planning and Meta-planning (MOLGEN: Part 2). *Artificial Intelligence,* 16, no. 2: 111-139.

Sussman, G. J., and G. L. Steele. 1979. Constraints- A language for Expressing Almost-Hierarchical Descriptions. *Artificial Intelligence,* 14, no. 1: 1-40.

Sutherland, I. 1963. "SKETCHPAD: A Man-Machine Graphical Communication System." Ph.D. dissertation, AI Laboratory, MIT.

Symbolics Inc. 1988. *Symbolics Common Lisp-Language Concepts.* Cambridge, MA: Symbolics Inc. 45-52.

Tanaka, T. 1983. Representation and Analysis of Electrical Circuits in a Deductive System. In *Proceedings of the IJCAI.* Los Altos: Morgan Kaufmann. 263-267.

Tanimoto, S. L. 1987. *The Elements of Artificial Intelligence.* Rockville, MD: Computer Science Press Inc.

Thathachar, M. A., and P. S. Sastry. 1986. Relaxation Labeling With Learning Automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-8, no. 2 (March): 256-267.

Thomas, S. J. Automated Construction of a Knowledge Base From Computer Aided Design Data. *NASA Kennedy Space Center Artificial Intelligence Section, Internal Technical Report,* KSC, FL: August 1987.

Ullman, J. R. 1982. Discrete Optimization by Relational Constraint Satisfaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-4. no. 5 (September): 544-551.

Vilian, M., and H. Kautz. 1986. Constraint Propagation Algorithms For Temporal Reasoning. In *Proceedings of AAAI,* Los Altos: Morgan Kaufmann. 377-382.

Waterman, D. A. 1986. *A Guide to Expert Systems.* Reading MA: Addison-Wesley.

Waterman, D. A., and F. Hayes-Roth. 1987. *Pattern-Directed Inference Systems.* New York: Academic Press.

Weiss, S. M., and C. A. Kulikowiski. 1984. *A Practical Guide to Designing Expert Systems.* Totowa, NJ: Rowman & Allenheld.

Williams, B. C. 1986. Doing time: Putting Qualitative Reasoning on Firmer Ground. In *Proceedings of AAAI,* 105-112.

Wos, L. 1985. What is Automated Reasoning? *Journal of Automated Reasoning* 1: 1-45.

Yamada, N., and H. Motoda. 1983. A Diagnosis Method of Dynamic System Using the Knowledge on System Description. In *Proceedings of IJCAI,* Los Altos: Morgan Kaufmann. 225-229.

Zadeh, L. A. 1965. Fuzzy Sets. *Information and Control,* 8: 338-353.

Zucker, S. W., Y. G. Leelerc, and J. L. Mohammed. 1981. Continuous Relaxation and Local Maxima Selection: Conditions for Equivalence. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-3. no. 2 (March): 117-129.